Assignment

PART 1 – Algorithms and Data Structure

[Total: 50 marks]

All code must be written by you, although you can use the lecture notes (and lab exercises), textbooks, and the Oracle Java website for guidance.

A) Extracting words from a text document is the first step for many text-based applications of artificial intelligence, e.g., detecting abusive tweets on Twitter. This task asks you to extract all valid words from the document **"Input219.txt"** based on a given vocabulary **"google-10000-english-no-swears.txt1"** (you can download both files from the Learning Central). Specifically, if a word token from **"Input219.txt"** matches a word in **"google-10000-english-no-swears.txt"** (case insensitively), you keep that word, otherwise you discard it. You should use *ArrayList* to store data within the Java program, so the return of this task is an *ArrayList* containing all valid words from **"Input219.txt"**. Note that these two text files are relatively big, you should consider how to make your program efficient.

[30 marks]
(Functionality: 16, Design: 10, Ease of use: 2, Presentation: 2)

B) Implement the merge sort (the pseudocode for merge sort is available in the lecture slides) in order to sort the words obtained in question (A) in alphabetical order, i.e., theoutput of your program will be the sorted words in alphabetical order. For the mergesort algorithm write a method e.g. *mergeSort(…)*, measure

- time that is needed to sort the first 100 of the words, first 200 of the words, first 300 of the words,etc. up to all words by the algorithm.
- count the moves/comparisons that occur while sorting elements

(Before attempting this exercise you should work through the Algorithms lab exercises, available on Learning Central. The techniques used there will help you to work out how to approach this part of the coursework, in particular there are examples on how to time algorithms and count the moves and swaps.)

[20 marks]
(Functionality: 12, Design: 4, Ease of use: 2, Presentation: 2)

[1] This vocabulary contains a list of the 10,000 most common English words in order of frequency of the Google'sTrillion Word Corpus.

PART 2 - Design and Implementation of Design Patterns

[Total: 50 marks]

Note: A list of files is included in the present coursework specification at the end for your convenience.

Download the ZIP file from Learning Central that contains the source code files of this task: **Product.java, ProductRecommender.java, ChoiceStrategy.java, MostFuturisticStrategy.java, MostPracticalStrategy.java.** This Java program creates two new products and then chooses between them, first by applying the **MostFuturisticStrategy** and then the **MostPracticalStrategy**. Futuristicness and practicality are both rated on a scale between 1(low) and 5 (high).

A) State the purpose of the *Strategy* design pattern. **[4 marks]**

B) Complete the implementation of the **MostFuturisticStrategy** class with a *chooseBetween* (Product a, Product b) method that returns Product a if its futuristicness is greater than or equal to the futuristicness of Product b; and returns Product b otherwise. **[8 marks]**

C) Complete the implementation of **MostPracticalStrategy** with a *chooseBetween* (Product a, Product b) method that returns Product a if its practicality is greater than or equal to the practicality of Product b; and returns Product b otherwise. **[8 marks]**

D) Modify the **doExample ()** method in the **ProductRecommender** class, so that each of the strategies can be applied, in order generate the following output:

```
Current strategy: choose most futuristic Chosen
vehicle: DeLorean DMC-12
Strategy changed: choose most practical
Chosen vehicle: LDV Maxus
```

[10 marks]

E) Draw a UML class diagram representing the resulting *Strategy* design pattern in this program. **[10 marks]**

F) Explain in your own words the role of each of the classes and the changes made in realising this design pattern as well the relationships between these classes and the interface. **[10 marks]**

Criteria for Assessment of PART 2

Credit will be awarded against the following specific criteria. Ability to:

- Understand the purpose of a Java program and apply principles of good object-oriented design in order to create a critique and/or modify the supplied programs (check the source code files provided);
- Apply and implement design patterns that pertain to a specified programming task;
- Reflect on the application of design patterns in OO Java programs.